Imperativ Programmering, tentamen

10 januari 2016

- Inga hjälpmedel (böcker eller annat material) är tillåtna.
- Jourhavande lärare: Martin Magnusson. Martin kommer förbi cirka kl 9 för att svara på eventuella frågor.
- Tentan har tre delar, som svarar mot betygskriterierna för betyg 3, 4 och 5.
  - För att bli godkänd, med betyg 3, behövs 20 av 32 poäng från ”3”-frågorna (fråga 1–6) **eller** 32 poäng totalt (från alla frågor).
  - För betyg 4 behövs dessutom 10 av 16 från ”4”-frågorna (fråga 7–10) **eller** 22 poäng tillsammans från fråga 7–12.
  - För betyg 5 behövs dessutom 10 av 16 poäng från ”5”-frågorna (fråga 11–12).

Betyg 3

Fråga 1

8 poäng

Hur många gånger skrivs bokstaven a ut i följande fall (förutsatt att koden exekveras i en `main`-funktion)?

1. ```
   for (int i = 0; i < 10; ++i)
   {
     printf("a");
   }
   ```

2. ```
   for (int i = 1; i < 10; ++i)
   {
     printf("a");
   }
   ```

3. ```
   for (int i = 0; i != 10; i++)
   {
     printf("a");
   }
   ```

4. ```
   for (int i = 0; i < 10; i == i+1 )
   {
     printf("a");
   }
   ```

5. ```
   int i = 0;
   while (i < 10)
   {
     printf("a");
   }
   ```

6. ```
   int i = 0;
   while (i < 10)
   {
     printf("a");
     i++;
   }
   ```

7. ```
   int i = 0;
   while (i = 1)
   {
     printf("a");
     i++;
   }
   ```

8. ```
   int i = 0;
   do
   {
     printf("a");
     i++;
   } while (i == 10);
   ```

Fråga 2

4 poäng

Fibonacci-talen är en talföljd där varje tal är summan av de båda föregående. Matematiskt kan talföljden definieras som

$$F(n) = \begin{cases} 0 & \text{om } n = 1 \\ 1 & \text{om } n = 2 \\ F(n-1) + F(n-2) & \text{annars.} \end{cases}$$

och de sju första talen är: 0, 1, 1, 2, 3, 5, 8.

(Ett exempel där Fibonaccital dyker upp är i ringarna i fjällen på talkottar.)

Skriv en funktion som tar ett tal n (positivt heltal) som argument, och skriver ut de n första Fibonacci-talen.

Du kan antingen skriva din funktion med C-syntax eller i klartext. Om du väljer att skriva i text är det viktigt att funktionsbeskrivningen ändå är lika detaljerad som den skulle behöva vara i ett C-program. Däremot behöver inte syntaxen vara enligt C. För den här uppgiften ges inga avdrag för syntaxfel, så länge det är klart vad som är meningen ska hända. Det är inte en övning i C-programmering, men däremot i (imperativt) algoritmiskt tänkande.

(Tips: ett sätt att lösa uppgiften är en loop som itererar n gånger och använder två variabler som håller reda på $F(n-1)$ och $F(n-2)$. Tänk i så fall på att inte råka skriva över någon variabel.)

Fråga 3

4 poäng

Datatyper, variabler och tilldelning.

1. *Varför* kan inte talet 314 lagras i en 8-bitars heltalsvariabel?

2. Vilket värde har `x` efter att följande C-kod är exekverad?

```
int a = 1;
int x = (a == 0);
```

3. Vilket värde har `x` efter att följande C-kod är exekverad?

```
float b = 3.1415;
int x = b;
```

4. Vilket värde har `x` efter att följande C-kod är exekverad?

```
int x = 0;
int y = 1;
{
  int x = 10;
  int y = 20;
  x += y;
}
x += y;
```

Fråga 4

8 poäng

Markera alla fel i följande program. Det finns både syntaxfel (som inte går igenom kompilatorn), och logiska fel (som ger fel resultat). Det är åtta fel totalt.

```
1   /* Ett program som beräknar "x upphöjt till y" för två positiva
2   heltal. */
3
4   #include <stdio.h>
5
6   power( int a; int b )
7   {
8     float result = 1
9     for {int i = 0; i <= b; ++i}
10      float result;
11      result = result * a;
12
13    return result;
14  }
15
16  int main()
17  {
18    int x = 10;
19    int y = 25;
20
21    // Beräkna x upphöjt till y
22    int z = power( x, y );
23
24    // Skriv ut resultatet
25    printf( "%d ^ %d = %d\n", x, y, result );
26
27    return 0;
28  }
```

Fråga 5

4 poäng

Skriv en C-funktion som givet ett nummer (1–7) för en veckodag skriver ut motsvarande veckodagsnamn (måndag–söndag). Använd ett fält (array) med strängar för att lagra veckodagsnamnen. (Det blir alltså ett fält av **char**-fält.)

Små syntaxfel ger inget poängavdrag i den här uppgiften, men det är viktigt att det framgår hur fält hanteras i C.

Fråga 6

4 poäng

Anta att du får en programbibliotek som innehåller följande funktionsdeklaration.

```
/**
Computes the area of a rectangle.
@param w Width of the rectangle.
@param h Height of the rectangle.
@return If w or h are negative, returns 0. Otherwise returns w*h.
*/
float rect_area( float w, float h );
```

Du har inte tillgång till funktionsdefinitionen, som ligger i en kompilerad objektfil. Ange ett antal testfall (minst 4) för att testa att funktionen verkligen gör det som sägs i kommentaren.

Betyg 4

Fråga 7

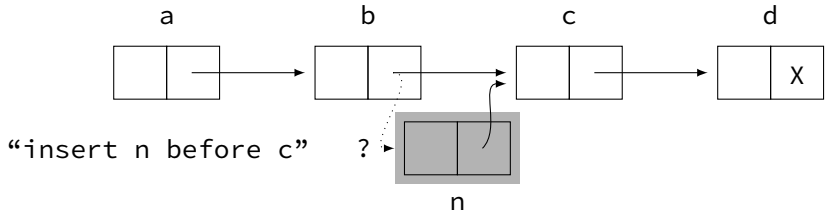
4 poäng

Vad är en abstrakt datatyp (ADT) och hur kan man implementera en ADT i C?

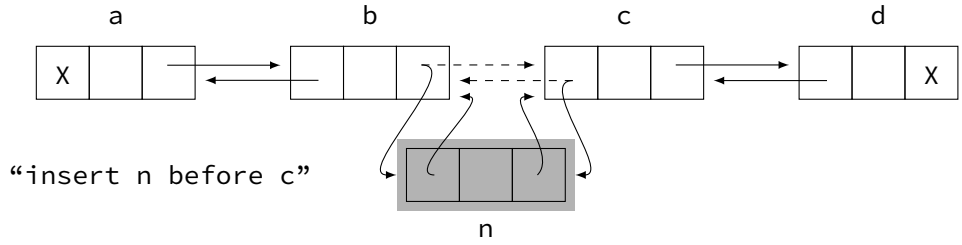
Fråga 8

Ett nackdel med länkade listor är att det inte utan vidare går att lägga till ett element *före* ett annat mitt i listan. Det beror på att varje element bara pekar på nästa. För att lägga in ett nytt element före element nr 3 i en lista med 4 element räcker det då inte att använda en pekare till element nr 3, eftersom också next-pekaren för element nr 2 måste uppdateras för att listan ska hänga ihop.

Bilden nedan visar vad som ska hända. I bilden markerar `X` en null-pekare, och det grå-markerade listelementet är det som ska infogas. Problemet är att det inte går att uppdatera pekaren i det andra elementet (markerad med en prickad pil och ett frågetecken) utan att också gå igenom listan från början för att hitta element nr 2.



Ett sätt att komma runt det är att använda en *dubbellänkad lista*, där varje element har en pekare både till det föregående och efterföljande elementet. Då är det lättare att hitta föregående element för att på så sätt uppdatera pekarna i listan efter att ha lagt till ett element i mitten. Se bilden nedan. Efter infogningen har pekarna uppdaterats. (De streckade pilarna visar på pekarna före infogningen, och de heldragna pilarna efter infogningen.)



2 poäng Skriv en **`struct`** som är en konkret implementation av en dubbellänkad lista.

3 poäng Skriv en C-funktion som lägger till ett listelement före ett annat, enligt bilden ovan.

Små syntaxfel ger inget poängavdrag i den här uppgiften, men det är viktigt att logiken stämmer, så att det tydligt framgår hur det är tänkt att datastrukturen ska uppdateras.

Fråga 9

2 poäng

Skriv en C-funktion som skriver ut ett fält (array) av heltal. Funktionen ska ta tre argument: ett fält med heltal, ett heltal `n` som anger antalet element, och en öppen textfil (av typen `FILE *`) som heltalen ska skrivas på.

Talen ska skrivas ut med ett mellanslag mellan varje tal. Utmatningen ska avslutas med en punkt och ett radslutstecken (`\n`). Godtyckligt stora fält ska kunna hanteras.

Exempel på utskrifter:

(a) `1 2 9 10 0 33 -6 2 .`

(b) `0 0 0 0 .`

(c) `.`

Här följer deklarationerna av några användbara biblioteksfunktioner.

```
FILE *fopen(const char *path, const char *mode);
int fclose(FILE *stream);
int printf(const char *format, ...);
int fprintf(FILE *stream, const char *format, ...);
int sprintf(char *str, const char *format, ...);
int fscanf(FILE *stream, const char *format, ...);
int sscanf(const char *str, const char *format, ...);
```

2 poäng

Beskriv vad som händer om `n` i själva verket är större respektive mindre än antalet tal som finns i fältet.

Fråga 10

3 poäng

För vart och ett av dessa numrerade påståendena om funktioner i C, vilket alternativ är sant? (Bara ett alternativ är sant i varje fall.)

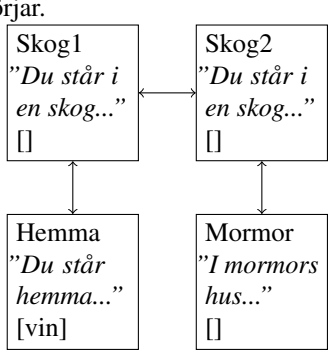
1. En funktion som *anropar sig själv*
   (a) får inte förekomma i C.
   (b) kallas konstruktor.
   (c) kallas iterativ.
   (d) kallas rekursiv.
2. En funktion som *innehåller en annan funktion*
   (a) får inte förekomma i C.
   (b) kallas konstruktor.
   (c) kallas iterativ.
   (d) kallas rekursiv.
3. En funktion i C
   (a) ger alltid ett värde av en bastyp (`int`, `char`, `void`, `struct` etc) som resultat.
   (b) ger alltid ett värde av en bastyp, *eller en pekare till en bastyp* som resultat.
   (c) ger antingen ett värde av en bastyp, eller en pekare till en bastyp, *eller ett fält (array) av bastyper* som resultat.

Betyg 5

En gång i tiden var textbaserade äventyrsspel populära. Ett typiskt exempel fungerar som följer.

Det finns en karta med platser, där spelaren kan gå mellan olika platser genom att gå åt väster, öster, norr eller söder. Varje plats har en beskrivning (i text) och eventuellt en lista med objekt som kan plockas upp eller användas. Spelaren har också en lista med objekt som hen har plockat upp. Spelet går ut på att gå runt i världen och lösa problem genom att använda olika saker med varandra, för att på så sätt lösa ett uppdrag.

Bilden nedan visar en liten spelvärld som motsvarar sagan om Rödluvan. Spelaren börjar i rutan ”Hemma”, där det ligger en flaska vin. Spelet går ut på att plocka upp vinet, gå genom skogen till mormors hus, och lämna vinet där. I de övriga tre platserna i världen finns inga saker när spelet börjar.



Spelet skulle kunna se ut så här under körning. Användarens inmatning är det som står efter >-tecknet.

```
Hemma.
Du står hemma. Din mamma har bett dig gå med en flaska vin till din
mormor. Ditt hus ligger i en tät skog, men det finns en stig som leder
norrut.
Det finns en flaska vin här.
> ta vin

Hemma.
Du står hemma. Din mamma har bett dig gå med en flaska vin till din
mormor. Ditt hus ligger i en tät skog, men det finns en stig som leder
norrut.
> gå norr

Skog1.
Du står i skogen. En stig leder söderut och österut. Du hör en varg
yla längre in i skogen.
> gå öst

Skog2.
Du står i skogen. En stig leder västerut och söderut. Mot söder ser du
att stigen leder fram till ett hus.
> gå syd

Mormor.
I mormors hus. Din mormor ligger sjuk i sängen.
> släpp vin

Mormor.
I mormors hus. Din mormor ligger sjuk i sängen.
Det finns en flaska vin här.

Du har klarat spelet!
```

Fråga 11

4 poäng

Nu ska du implementera en del av ett sådant här textspel.

Skriv en C-implementation för de datatyper som behövs för världen och spelaren. Du behöver inte tänka på att separera implementation och specifikation här, men definiera hur du i C kan representera en värld enligt exemplet ovan samt en spelperson. Spelet måste alltså kunna veta var spelaren är. Spelet måste också kunna veta hur de olika platserna hänger ihop (åt vilka håll man kan gå från varje plats) och beskrivningen för varje plats. Du behöver inte implementera någon datatyp för att hålla reda på vilka saker som finns på olika ställen, utan bara hur platserna hänger ihop, hur deras beskrivningar lagras, och hur spelet vet var spelaren är.

Ange vilka antaganden du gör om du behöver göra antaganden som inte är specificerade i de här instruktionerna.

2 poäng

Skriv också en funktion (som använder dina datatyper) som flyttar spelaren från en plats till en annan, om det är möjligt. Alltså den funktion som ansvarar för att spelaren flyttas från ”Hemma” till ”Skog1” om hen går norrut från ”Hemma”.

4 poäng

Diskutera också två eller fler alternativa sätt att lagra vilken plats spelaren är på. Hör det till datatypen för spelaren, världen, eller ska det ligga någon annanstans? Vad har det för för- och nackdelar?

4 poäng

Diskutera också två eller fler alternativa sätt att lagra textsträngarna för varje plats. Vad har de för för- och nackdelar?

Fråga 12

2 poäng

För var och en av raderna 1–4 i koden nedan, vilket av alternativen (a)–(b) stämmer?

```
1 int * x1 = malloc( 10 * sizeof( int ) );
2 int * x2 = malloc( 10 );
3 int x3[10];
4 int * x4 = 10;
```

(a) Allokerar plats för 10 `int`.

(b) Allokerar plats för 10 bytes.

(c) Allokerar plats för en pekare.

(d) Allokerar plats för en pekare och 10 `int`.

(e) Allokerar plats för en pekare och 10 bytes.

(f) Allokerar plats för en `int`.

(g) Allokerar plats för 10 `int`.